Comparisons of Features for Automatic Speaker Verification

Ziqi Wang, Qiong Hu, Yu-Chun Hung

University of California, Los Angeles

wangzq312@g.ucla.edu, junesirius@ucla.edu, meteors121@gmail.com

Abstract

Automatic Speaker Veification (ASV) techniques are widely used in our daily life, ranging from telephone banking to voice voice assistance authentication. In this work, we build an ASV system using k-nearest-neighbour (KNN) classifier and compared a variety of features, including source-filter model based features, cepstral coefficients (CC) features, neural network (NN) based features and i-Vector based features. We also further improved the system performance using score fusion techniques. The system is meant to be text-independent and also be resilient to the mismatch in speech styles, e.g. phone call speech and sentence reading speech. Our system achieves about 4 times verification performance improvement in matched-style speech trials compared to the baseline built with pitch features, and approximately 2 times better performance in mismatch scenarios.

1. Introduction

In the domain of speaker recognition, there are two relative but separate tasks: speaker recognition and speaker verification. In speaker recognition, the task is to identify an unknown speaker from a set of known speakers; while in speaker verification, the task is to compare two speech utterances and decide whether they were spoken by the same speaker. In this paper, we are going to focus on the task of speaker verification [1].

Automatic speaker verification (ASV) technique plays an important role in the modern industry such as many e-commerce applications, general business interactions, forensics, and law enforcement [1]. For example, the technique can be used by a bank to decide whether an unknown audio sample and a reference sample of a potential customer are spoken by the same person, and decide whether to continue a voice-activated transaction on telephone.

While human ears have been trained to be good at identifying and verifying speakers from utterances under difference situations, the task could be challenging for the computer because of but not limited to the following reasons:

- The same speaker does not always say the same words in a same way, which is known as style shifting or intraspeaker variability [2]. The speech utterance may have a shifted pitch, loudness and style when the speaker is in a difference environment, mood, health circumstance, et al.
- Various recording devices and transmission approaches make the problem even more difficult. For example, due to the sample rate and transmission limits, the same speaker may not sound the same even for human ears, let alone the machines.
- Environment noises and unvoiced segments in the utterance would decrease the signal-noise-ratio, which may also exacerbate the problem.

Due to the above reasons, we conducted the research trying to find the effective features and feature extraction methods that could verify speakers as accurately as possible. Our system workflow is shown in Figure 1. Note that the K-Nearest Neighbour is the only backend classifier we use to make decision on speech pairs, and all the experiments we do lies in the feature extraction part.

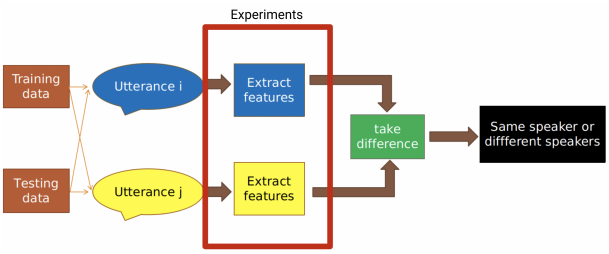

Figure 1: *Our ASV System Pipeline*

Our contributions are listed as below:

1. Tested on different source filter model based features, including LPC, LPCC, and VQual Feature Set.
2. Looked into the influence of PCA, Delta-coefficients and the number of coefficients in Cepstral Coefficient based feature extraction approaches, including LFCC, MFCC and CQCC
3. Tried Neural Network embedding based features using neural networks pretrained on large datasets.
4. Tried GMM-UBM based i-vector methods and its behavior with different variable sets and training sets.
5. Used model fusion method to come up with a final fused model that is able to combine the advantages of different feature extraction methods above and have a relatively good performance on the test sets and blind sets.

2. Related Work

Researches on Automatic Speaker Verification started early in 1970s [3]. In these early works, statistical features like pitch, low formant frequencies and low-frequency intensities are used primarily. Cepstral Coefficients(CC's) based features are introduced in [4], where CC's are calculated with Linear Predication Coding (LPC) analysis, and passed though a pipeline including frequency distortion correction and time warping to calculate the distance. In 2000, MIT Lincoin Laboratory introduced a Gaussian Mixture Model (GMM) based ASV system [5]. In such method, a universal background model (UBM) is trained using GMMs with a large corpus, and Relevance Maximum A Posteriori (MAP) is then applied to derive supervectors of individual speakers. GMM-UBM models have a huge impact in ASV area, and was succeeded by Joint Factor Analysis [6] and

i-Vector [7] method. The way that the speaker supervector differs from universal supervector of UBM may contain information both from the channel and the speaker. JFA attempts to separate the those two contribution factors and estimate the speaker identity, while i-Vector maps the information of the speaker and the channel in the supervector into a lower dimension "total variability space", and then use linear discriminate analysis (LDA) to further remap the i-vector onto a new basis that best separate the speakers. The extractionn of i-vector usually take a long utterance from each speaker.

Since 2012, new speaker verification methods have been inspired by the breakthroughs in neural networks. Some of theses work attempts to replace GMM with Deep Neural Networks to obtain vectors for speech frames. In 2014, Variani et al. proposed d-vector [8], which take the energy features from a filter bank given a speech sample, and use the activation of the last hidden layer as the speaker embedding. It claimed to improve the Equal Error Rate(EER) of ASV systems by 14%. J-vector [9] is proposed in a similar way and use Probabilistic Linear Discriminant Analysis is then used as backends for ASV tasks. Both d-vector and j-vector are targeting at speech dependent ASV system, where the content of speech is fixed. [10] proposes x-vector, in which the features are extracted frame-by-frame using a frame-level subnetwork and then merged with a statistics pooling layer. Then speaker embedding is extracted using the activation of the following layers. In recent years, mechanisms like attention-based method [11] and Generative Adversarial Networks (GANs) [12] have also been used in ASV area. Meanwhile, some researchers noticed the threats that computer generated voice and pre-recorded speech pose to ASV system and starts to work in the area of ASV anti-spoofing [13] [14].

3. Proposed Methods

3.1. Source Filter Model Based Features

In source-filter model, human voice is considered to be a convolution of human voice source excitation (voiced pitch and unvoiced noise) and human vocal tract formulation.

Inspired by this model, we use the a_k coefficients in Linear Predication Coding (LPC) analysis as a feature vector. It is shown in literature [3] that the low-order formants can better represent speaker identity. Also, one major difference between phone call speech and reading speech recordings is the sampling rate. Thus we firstly down-sample the original speech frames to 8kHz, and then take 8-order LPC to calculate the coefficient features. The final feature vector is a mean average of the LPC a_k coefficients across speech frames to represent vocal tract filter features. The average pitch of this utterance is also added to the speech vector to incorporate the source feature.

We also tried to employ time-averaged Linear Prediction Cepstral Coefficients (LPCC) as a feature vector, which is based on the LPC analysis. We will not discuss too much details of LPCC. In our implementation, we used a 48-order LPC on the original speech waveform and then convert it to 10 cepstral coefficients using [15].

The second feature set we use in the section is VQual feature set [16]. This feature set includes includes both source- and filter-related features that are critical to "the timbre of voice". We select the combination of a variety of features (averaged over time) to form the final feature vector, including H2K, H5K, H1H2c, H2H4c, H1A1c, H1A2c, H1A3c, H2KH5Kc, H42Kc, F2K, CPP, pF0, pF1, pF2, pF3, SHR, strF0. The features above including the amplitude of the x-th harmonics (Hx), Cepstral Peak Prominence (CPP), Subharmonic to Harmonic Ratio (SHR), etc. For details, please refer to the VoiceSauce [17] tool manual.

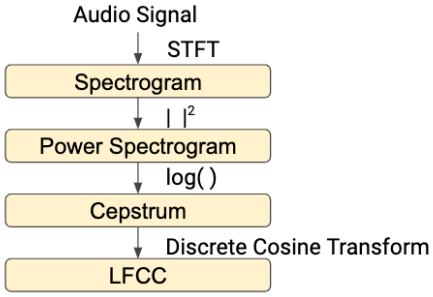

Figure 2: *Calculation Pipeline off Linear Frequency Cepstral Coefficients*

3.2. Cepstral Coefficient Based Features

In this section, we used time-averaged Cepstral Coefficients (CC's) as the feature vector. We experimented on three different types of CC features: Linear Frequency Cepstral Coefficients (LFCC), Mel Frequency Cepstral Coefficients (MFCC) and Constant-Q Cepstral Coefficients (CQCC).

The calculation of LFCC is the "standard approach" (see Figure 2), where the audio is analyized through Short-Time Fourier Transform (STFT) to get the spectrogram. Then we take the logarithm on the amplitude to the spectrogram to acquire cepstrum, and finally use Discrete Cosine Transform (DCT) to convert it to CC's. The basic idea behind CC methods is still source-filter model, where the source and vocal filter are multiplicative in the frequency domain. The logarithm in LFCC converts this relationship a additive one and helps to separate and model the source and filters separately.

MFCC is widely used in speaker recognition and speech recognition field as a feature. Mel-frequency analysis is based on research in speech perception field. It maps the frequency domain speech into a mel domain that is closer to the human ear perception of frequencies. In MFCC, this mapping happens after the spectrogram is calculated and is implemented using a set of triangular-shaped mel-filters.

We also make attemps with CQCC proposed in [18]. The key idea of CQCC is to substitue the STFT with Constant Q Transform(CQT) in the standard CC pipeline (Figure 2). Q-factor is defined as the central frequency divided by bandwidth, i.e.,

$$Q = \frac{f_0}{Bandwidth} \tag{1}$$

Figure 3 [1] demonstrates a comparison between CQT and STFT, where each red dot is the intersection between the center of sampling window in the time domain and the central frequency of a filter in the frequency domain. The blue dotted lines indicates the boundary of the filters, and the space between boundaries is the filter bandwidth. It is clear that, by maintaining a constant Q-factor, we can achieve a higher frequency resolution in low-frequency bands, and a higher time-resolution when frequencies go higher. This might be helpful for computers to perceive speech signal and better extract the identity-related information.

[1] This figure is directly imported from [18]

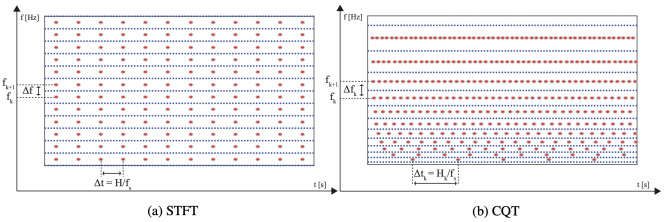

Figure 3: *Comparison of STFT and CQT.*

A Common hyper-parameter in CC methods is the number of coefficients to include. In the experiments, we investigate the effects of the number of coefficients have on ASV performance, using vanilla LFCC, MFCC and CQCC.

Also, $\Delta - CCs$ and $\Delta^2 - CCs$ are the first and second order derivatives of cepstral coefficients. They are capable of capturing the dynamics in speech. It is also possible that including these Δs may help improve performance.

Another observation is that many features have very high dimensions. For example in LFCC, if we use 450 filters in frequency domain and also include Δs, we will have a 1350 dimension feature. Since the KNN model is the only back-end we are using, too high dimensions make the speaker representation more entangled (in other words, curse of dimensionality). So it important that we reduce features dimensions and make feature dimensions orthogonal, while keeping the original feature information or variance. We found that Principle Component Analysis (PCA) is perfect for this task.

Suppose we have m samples and n features to form a matrix $\mathbf{X} \in R^{m \times n}$, what PCA does is to find a mapping $T : \mathbf{X} \to \mathbf{X}' \in R^{m \times n}$. T projects the original n dimensions onto a new space expanded by n orthogonal basis, and makes sure that the projection distance is minimized. In the new matrix $\mathbf{X}'$, the n feature dimensions are sorted in descend order of "importance", thus we can discard a few least important features to reduce the dimension with mapping $T' : \mathbf{X} \to \hat{\mathbf{X}} \in R^{m \times n'}$, $n' < n$.

We then perform a grid search over the use of deltas and PCA to explore whether they individually or jointly help ASV performance. For each of the CC's listed above, we compare the ASV performance with and without the Δs, and use PCA to lower the data dimensions to different levels.

3.3. Neural Network-Based Features

As we discussed in related work, neural networks have achieved great success in ASV tasks. A typical network structure for the neural network-based ASV system takes the form of a Siamese Network. The input of a siamese network are two audio clips x_1 and x_2 , and they are passed through two identical subnetworks with shared weights to produce feature vectors $\mathbf{F}_1$ and $\mathbf{F}_2$. The feature vectors are then passed thought a classifier subnetwork to determine if the utterances come from the same speaker. Since KNN is the only allowed backend in this project, we, in turn, direct use the $\mathbf{F}_1$ and $\mathbf{F}_2$ as the feature vector of our system.

One significant challenge here is that neural networks must be trained with a rich dataset, while our project has only a small dataset. Our solution is to apply neural networks whose weights trained on large datasets and directly apply to our dataset. We selected two models from [19] [20], whose model structure is shown in Figure 4[2]. On the left is a deep convolution neural network (DCNN) structure, and on the right is a deep residual network structure with skip connections at the beginning and the end of the network to enable better gradient backpropagation.

Layer	Support	Filt dim.	# filts.	Stride	Data size
conv1	7×7	1	96	2×2	254×148
mpool1	3×3	-	-	2×2	126×73
conv2	5×5	96	256	2×2	62×36
mpool2	3×3	-	-	2×2	30×17
conv3	3×3	256	384	1×1	30×17
conv4	3×3	384	256	1×1	30×17
conv5	3×3	256	256	1×1	30×17
mpool5	5×3	-	-	3×2	9×8
fc6	9×1	256	4096	1×1	1×8
apool6	1×n	-	-	1×1	1×1
fc7	1×1	4096	1024	1×1	1×1
fc8	1×1	1024	1251	1×1	1×1

res-50
7 × 7, 64, stride 2
3 × 3, max pool, stride 2
$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$
$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$
$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
9 × 1, 2048, stride 1
1 × N, avg pool, stride 1
1 × 1, 5994

Figure 4: *Neural network structures in [19] and [20]*

These models are trained on VoxCeleb Dataset [21]. VoxCeleb consists of over 1 million short clips of human speech extracted from interview videos on YouTube and has more than 7000 speakers. According to [20], the models are firstly pretrained with a speaker recognition loss, whose last layer is about 6000 dimensions, and each node represents a single speaker. Then the final layer was replaced with a 128 dimension layer, and the parameters are further fine-tuned with speaker verification task using contrastive loss. Pretrained models from the authors are available on Github [22].

We compare the performance of using the final layer of these two models as feature embedding for speech utterance. The feature dimension is 1024 for DCNN model and 128 for ResNet-50 model. We also try two take speech pairs, calculate their embeddings used the above method, and finally use the euclidean distance between the two embeddings as a one-dimension feature. Among those attempts, ResNet-50 128 dimension feature has the best ASV performance. For details, please refer to the evaluation section.

3.4. GMM-UBM and i-Vector

A Gaussian Mixture Model (GMM) is a linear superposition of Gaussian probability density functions, with a number of mean vectors, covariance vectors, and weights of each individual mixture components [1]. The formula of a vector x modeled by K multivariate Gaussian components with mean vectors μ_k, covariance matrices Σ_k, weight w_k,where $k = 1, 2, \ldots, K$, is given by

$$f(x) = \sum_{k=1}^{K} w_k \mathcal{N}\left(x \,|\mu_k, \Sigma_k\right), \tag{2}$$

where the formula of the multivariate Gaussian is as follows,

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}. \tag{3}$$

The GMM approach has been effective in speaker recognition tasks, but for speaker verification, an alternative speaker model, known as *universal background model* (UBM) is needed. The UBM is trained from a large dataset with tremendous number of speakers so that it could be considered as speaker-independent, or "universal".

[2]The content of this figure is direction imported from [19] and [20]

Based on GMM-UBM, the i-vector approach is proposed [23], and the principle formula is represented by

$$s = m + T\omega, \tag{4}$$

where s is the speaker- and channel- dependent GMM supervector, m is the UBM's mean supervector, T is called total variability matrix, which can be estimated by the expectation-maximization (EM) algorithm [24], and ω is the i-vector that extract speaker recognition features.

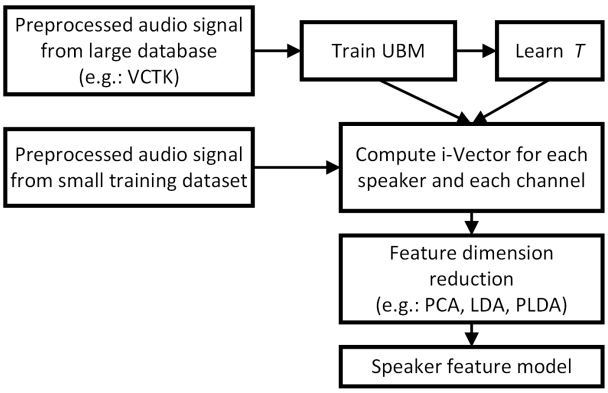

Figure 5: *Implementation pipeline of the i-vector method*

In the implementation as seen in Figure 5, we also applied Linear Discriminant Analysis (LDA) to reduce the dimension of the i-vector [25], and removed the unvoiced segments based on the short-time energy threshold during audio preprocessing. The resulting reduced i-vector is considered as the feature for each speech utterance.

3.5. Model Fusion

In this section, we leverage fusion techniques trying to fuse multiple models in order to yield better overall performance.

The first attempt is **feature concatenation**. The intuition is that different features might pay attention to different aspects and details of a speech utterance, thus they might be complementary. After extracting features, we concatenate them to form a larger feature vector. We firstly try to fuse MFCC feature with LPC a_K's. Secondly, we fuse the MFCC with VQual features sets as they are meant to be a compliment to MFCC features [16].

The second attempt is **score-level fusion**. The score-level fusion happens after we run all the model and got the test scores. The fused score is obtained by normalizing all the individual scores and calculating a weighted average. We assign the weights based on signal model performance.

4. Evaluation

4.1. Data and Evaluation Matrices

The evaluation of this project use a subset of the UCLA Speaker Variability Database developed by UCLA SPAPL. It contains two-second utterances by 50 different male speakers sampled at 22.05 kHz. All the utterances are text-independent, i.e, all speakers are speaking different text. There are utterances from 20 different speakers in the test set with the same structure. Those speeches have two different styles: read speech and phone call conversation. Thus there will be matched-style pairs and mismatched pairs, and our system is trained an tested in the following settings.

In on case, we use the train on read-read pairs and test with 1) read-read, 2) phone call - phone call and 3) read - phone call. In a second case, we train on phone call-phone call pairs and again test with 1) read-read, 2) phone call - phone call, and 3) read-phone call separately. In summary, the system will be trained on matched-style data and tested on mismatched data, which is a challenge to the generalization ability of the feature extraction schemes.

The only metric we use for the evaluation of ASV system performance is Equal Error Rate (EER). It is define as the rate where the false acceptance rate of the system and its false rejection rate equal to each other. It is widely used to evaluate ASV tasks as accuracy does not make sense when dealing with biased data (negative labels for most attempts).

4.2. Source Filter Model Based Features

Using both the *pitch and LPC* features, *LPCC* features and the *VQual feature set*, we do not get a very satisfying result. None of the models can get an get an EER under 30% for matched testing cases. The best feature among theses three is LPCC features. When trained on read-read pairs, the testing EER is 29.75% for read-read cases and 41.2% for mismatched cases. Thus we determine that using these feature along cannot build very trustworthy ASV system as they only improve the simple baseline built with average pitch marginally. We thus do not include the result of these models in the final result table.

4.3. EER Versus Number of Coefficients

For the LFCC, MFCC, and CQCC model, we investigated the relationship between EER and the number of coefficients without using PCA or Δ-coefficients. The result is plotted in figure 6. The x-axis is the number of coefficients and the y-axis is the EER. From the figure, we can see that the best number of coefficients is 511 for LFCC, 33 for MFCC, and 101 for CQCC.

4.4. EER Versus PCA Strength and Delta

In these section we wish to answer two questions. Does adding PCA help? Does adding Δ-coefficients help? For the LFCC, MFCC, and CQCC model, we tried different "PCA strength", and different Δ settings.

In MATLAB implementation, the "importance" of feature dimensions are given by the return vector $latent$(principal component variances) that is the eigenvalues of the co-variance matrix of X. The values of $latent$ are sorted in descending order and sum up to 1. We use the cumulative sum of the latent values of the dimensions we keep as a indication of "PCA strength". For example, if we keep n_1 dimensions whose latent sums up to 0.9999, and n_2 whose latent summation is 0.99999, then we know $n_2 > n_1$.

The result is plotted as figure 7. The x-axis is the PCA strength, the y-axis is the EER, and the different color is for adding different Δs – blue for no Delta, red for adding $Delta$, and yellow for adding both Δ and Δ^2.

From the LFCC plot in Figure 7, we can see that adding PCA helps decrease EER, and the best $latent$ is 0.99999. Within each $latent$, adding $Delta$s sometimes helps but only have marginal improvements.

From the MFCC plot in Figure 7, we observed that the EER increases as we increase the PCA strength. The model has best performance when there is no PCA. Within each $latent$, adding

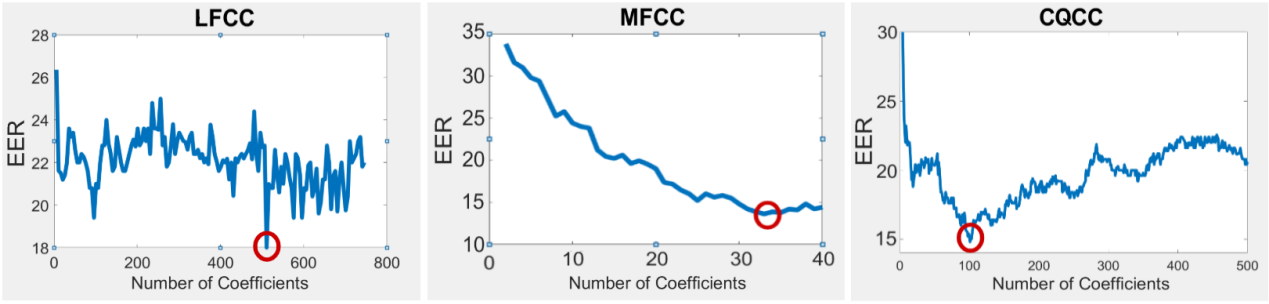

Figure 6: *EER versus Number of Coefficients for LFCC, MFCC, and CQCC models.*

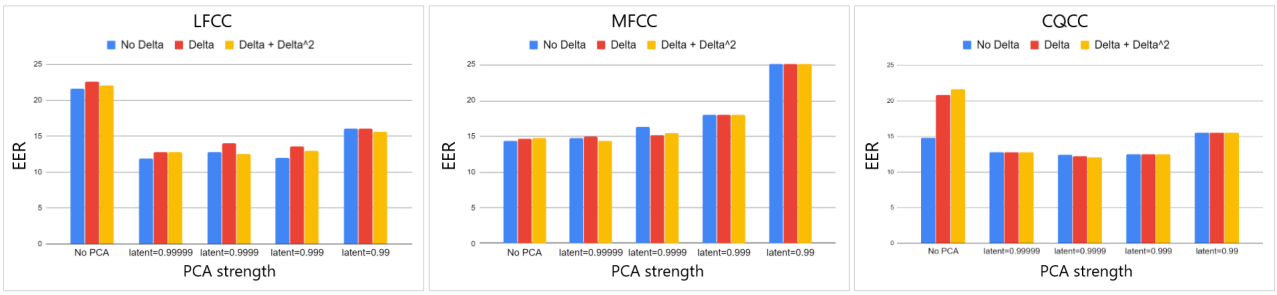

Figure 7: *EER versus different PCA strength and deltas for LFCC, MFCC, and CQCC models.*

$Delta$s sometimes decreases EER but the result doesn't have much difference.

From the CQCC plot in Figure 7, we know that adding PCA helps decrease EER, and the best $latent$ is 0.9999. For each $latent$, adding $Delta$s does not have much improvement on the EER. Also note that, when we did not use PCA, adding Delta highly increases EER for this CQCC model.

To sum up, PCA is not suitable for MFCC, but with proper value it may improve EER for LFCC and CQCC. When there is no PCA, adding Delta almost always makes the model worse, while adding Delta doesn't make much difference when PCA is utilized.

4.5. Best Result of Each Model

We tune the hyperperameters for each models to get the best settings. The feature for the baseline model is using only the pitch of each speaker. The result is showed in Table 1. We list the configurations where the best results are acquired in Table 2.

The LFCC, MFCC, and CQCC models all achieve EER lower than 15% for Read-Read testing set, and have EER between 15% to 20% for Phone-Phone testing set. However, the EER is high for the mismatched cases, and couldn't get under 35%.

The result for the i-Vector model has considerably great performance when the UBM and speaker model are both trained and tested on the same given dataset, as long as the dimensions are large enough. For our best result, when the number of mixture for GMM is 1024, and the dimension of i-vector is 400 [26], all the EERs in the evaluation matrices remain below 10%. However, a UBM model should be trained on a large dataset for good generalization ability, and our toy experiment also shows that this model trained from limited speaker dpes not generalize well for unseen speakers, and thus do not include it in the further consideration for blind tests as well as the fusion model.

Among the two neural networks, the ResNet structure with 128 dimension final layer embedding outperforms the DCNN structure, which is not surprising. The 128 dimension model shows a balanced performance in both matched and mismatch scenarios, with EER ranging from 16.14% to 23%.

The feature concatenation model has very similar result compared with the MFCC-based model, but it is able to get a slightly lower EER for the mismatched cases. However, this improvement is only marginal.

Apart from the i-Vector model, the score fusion model has the lowest EER for the Read-Read and Phone-Phone matched cases, and the Neural Network model has the lowest EER for the mismatched cases. In addition, both of the model have balanced results.

Therefore, we conclude that the score fusion is out best model, since it achieved EER about or below 10%, and it has balanced result for both matched and mismatched cases.

5. Discussions and Future Works

After all the experiments and comparisons, we found out that the source filter model based features failed to effectively extract speaker-dependent features, and only had a slight improvement compared to the baseline. The reason probably is that information only extracted from formant frequencies, pitch, et al are not able to fully describe a speech's timbre, and statistical features suffers from speech style variation. For example, due to the telephone sample rate and channel frequency response, a person can sound very differently in terms of timbre. Therefore, the features including *pitch and LPC*, *LPCC*, and *VQual feature set* alone had a poorer performance compared to other methods for speaker verification tasks.

As for cepstral coefficient based features, including LFCC, MFCC, and CQCC, we found out that MFCC had a slightlyy

Training Set	Read-Read			Phone-Phone		
Testing Set	Read-Read	Phone-Phone	Read-Phone	Read-Read	Phone-Phone	Read-Phone
Baseline	39.4	40.77	45.2	39	40.89	44.8
LPCC	29.75	26.8	41.2	30.8	27.6	40.6
LFCC	11.85	20.2	41.8	13.4	19.4	41.2
MFCC	13.6	16	35.6	16	17.6	36.25
CQCC	12.8	19.2	42.39	14.6	17.79	41.98
NN	16.14	17.49	**23**	18.4	17.6	**22.31**
i-vector	5.24	8.65	8.58	5.86	8.32	8.16
Feature Concat	13.6	17.02	33.68	17.2	17.6	35
Score Fusion	**9.28**	**10.17**	24.2	**10.8**	**10**	23.4

Table 1: *Best Result of Each Model*

Feature Name	**Settings**	**Use Δs**	**Use PCA**
LPCC	48 LPCs converted to 10 coefficients	No	No
LFCC	450 Coefficients	No	0.99999
MFCC	33 Coefficients	No	No
CQCC	101 Coefficients	Δ and Δ^2	0.9999
NN	ResNet-50 Model, Last hiddle layer 128 dimension embeddings	-	No
i-Vector	13 Coefficients MFCC, 1024 GMMs and 400 dimension i-Vector, LDA applied	-	No
Feature Concat	33 Coefficients MFCC	No	No
	F0, F1, F2, F3, H1-H2, H2-H4, H4-H2k, CPP in VQual	-	No
Fusion	50% NN ResNet-50 Model, Last hiddle layer 128 dimension embeddings	-	No
	25% MFCC 40 Coefficients model	Δ and Δ^2	0.99999
	25% LFCC 450 Coefficients model	No	0.99999

Table 2: *Corresponding Settings to Generate Results in Table 2*

better performance, which probably because it processed the audio information in a way similar to human ear perception of frequencies. When looking into the influence of PCA and Deltas for these three methods, our experiments show that Deltas almost always make the model indifferent or worse, and PCA only improve EER for LFCC and CQCC if it has proper values. The reason is probably that Deltas emphasize the dynamics in the utterance, which is more sensitive and appropriate for speech content recognition instead of content-independent tasks like speaker verification. PCA mainly works on reducing the feature dimension to extract features that truly matter, it cannot recover the information that has already been lost in previous pre-processing steps. It is only effective if the original feature dimensions has strong correlations or redundancy. The limitation of these models is that it almost always have a trade-off between the cases where training and testing set are matched or mismatched. For example, when the performance on matched cases improves, showing that the model extract more case-dependent features, the performance on mismatched cases get worse, vice versa.

In the GMM-UBM based i-vector methods, we found out that the EER test results have a tremendous improvement when increasing the number of mixtures for GMM and the dimension of i-vector. However, due to the limitation in universal speaker model and computation cost, we only trained UBM on a) VCTK open dataset (103 speakers, 420 utterance of 2 seconds for each speaker) with 512 mixtures and 200 dimension of i-vector, b) given training dataset (70 speakers, 10 utterance of 2 seconds for each speaker). In the first case, the EER for testing set are generally around 30%, and for the model in the second case, the EER can be reduced to be below 10%, but have a really poor performance when tested on unseen speakers (around 40%). The experiments show that the i-vector would be more suitable for speaker recognition tasks, instead of speaker verification tasks. Therefore, we did not include it in the final fusion model for blind tests.

The Neural Network based model has a more balanced performance between matched and mismatched cases, and generally have the best balanced performance, which is why it takes the largest proportion in the final fusion model. However, because of the uninterpretability of most of the deep learning model, we still could not fully understand what is the most important or intuitive features that human ears utilize to distinguish different speakers from different speech utterances. Also, limited by time and data availability, we directly applied the model trained on another dataset without using any transfer learning techniques. In the future, it is possible to fine-tune the model parameters on our dataset to achieve a smoother model transfer.

As a result, considering that NN is generally the best single model for all the tests in the evaluation matrices, MFLL and LFCC have a slight better performance in the matched scenarios, we used a ratio of 50% NN + 25% MFCC + 25% LFCC as a final fusion model for the speaker verification task. The results show that it achieved EER near 10% for matched cases and about 20% for mismatched cases.

For future works, there are many more questions that worth asking and looking into, for example, how to change the structure of Neural Networks to extract better features, how to understand and interpret the speaker features extracted from Neural Networks, how to train the UBM so that the i-vector methods can be transferable, is there other methods that can solve the speaker verification problem, can methods for speaker recognition and speaker verification be transferable, et al. The domain of speaker recognition still remains ongoingly challenging in many ways, and we would be glad if our work can uncover the mystery at least a little bit.

6. References

[1] J. H. Hansen and T. Hasan, "Speaker recognition by machines and humans: A tutorial review," *IEEE Signal processing magazine*, vol. 32, no. 6, pp. 74–99, 2015.

[2] P. Eckert and J. R. Rickford, *Style and sociolinguistic variation*. Cambridge University Press, 2001.

[3] R. Lummis, "Speaker verification by computer using speech intensity for temporal registration," *IEEE Transactions on Audio and Electroacoustics*, vol. 21, no. 2, pp. 80–89, 1973.

[4] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 2, pp. 254–272, 1981.

[5] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[6] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel, "A study of interspeaker variability in speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 5, pp. 980–988, 2008.

[7] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[8] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4052–4056.

[9] N. Chen, Y. Qian, and K. Yu, "Multi-task learning for text-dependent speaker verification," in *Sixteenth annual conference of the international speech communication association*, 2015.

[10] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification." in *Interspeech*, 2017, pp. 999–1003.

[11] F. R. rahman Chowdhury, Q. Wang, I. L. Moreno, and L. Wan, "Attention-based models for text-dependent speaker verification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5359–5363.

[12] G. Bhattacharya, J. Monteiro, J. Alam, and P. Kenny, "Generative adversarial speaker embedding networks for domain robust end-to-end speaker verification," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6226–6230.

[13] M. Alzantot, Z. Wang, and M. B. Srivastava, "Deep residual neural networks for audio spoofing detection," *arXiv preprint arXiv:1907.00501*, 2019.

[14] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen, and K. A. Lee, "Asvspoof 2019: Future horizons in spoofed and fake audio detection," *arXiv preprint arXiv:1904.05441*, 2019.

[15] "convert linear prediction coefficients to cepstral coefficients - matlab." [Online]. Available: https://www.mathworks.com/help/dsp/ref/dsp.lpctocepstral-system-object.html

[16] S. J. Park, C. Sigouin, J. Kreiman, P. A. Keating, J. Guo, G. Yeung, F.-Y. Kuo, and A. Alwan, "Speaker identity and voice quality: Modeling human responses and automatic speaker recognition." in *Interspeech*, 2016, pp. 1044–1048.

[17] "A program for voice analysis." [Online]. Available: http://www.phonetics.ucla.edu/voicesauce/

[18] M. Todisco, H. Delgado, and N. Evans, "Constant q cepstral coefficients: A spoofing countermeasure for automatic speaker verification," *Computer Speech & Language*, vol. 45, pp. 516–535, 2017.

[19] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[20] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[21] "Voxceleb dataset." [Online]. Available: http://www.robots.ox.ac.uk/ vgg/data/voxceleb/

[22] A-Nagrani, "a-nagrani/vggvox," Feb 2019. [Online]. Available: https://github.com/a-nagrani/VGGVox

[23] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Tenth Annual conference of the international speech communication association*, 2009.

[24] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[25] S. J. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[26] Y. Liu, Y. Tian, L. He, and J. Liu, "Investigating various diarization algorithms for speaker in the wild (sitw) speaker recognition challenge." in *Interspeech*, 2016, pp. 853–857.